ECE 201 Quiz

Based on Lecture 1, Lecture 2

Section 1: Conceptual Understanding

Q1. What is software in the context of a computer system?

Q2. Define machine language. Why is it considered the only language understood by computer hardware?

Q3. What is a source program and how is it transformed into an executable program?

Q4. Explain the role of a compiler in program development.

Q5. What is the purpose of the `main()` function in a C program?

Q6. Define a preprocessor directive and give an example.

Q7. What is the difference between machine language and high-level language?

Q8. What does the `#include <stdio.h>` directive do in a C program?

Q9. What are comments in C, and how do block comments differ from line comments?

Q10. Describe the structure of a basic C program using labeled components (e.g., declarations, statements).

Q11. What does the computer system consist of?

Q12. What are the main components of a computer hardware?

Q13. Define computer software and give two examples.

Q14. What is the role of a compiler in program execution?

Q15. Arrange the following in chronological order: Machine Language, Assembly Language, High-Level Language.

Q16. What are the advantages of high-level languages over low-level languages?

Q17. What is the difference between a source program and an object program?

Q18. What are the six types of C tokens?

Q19. Which of the following are valid identifiers?

- _sum
- 2value
- total_marks
- sum-salary
- INT
- student name

Answer:

Q20. What is the significance of the keyword `int` in C?

Q21. What is the difference between a keyword and an identifier?

Q22. What does it mean that C is case-sensitive?

Q23. What is the difference between signed and unsigned integers?

Q24. What is the precision of `float`, `double`, and `long double`?

Q25. What is the syntax for declaring multiple variables of the same type?

Q26. What is the difference between a variable and a constant?

Q27. What is the difference between a character constant and a string constant?

Q28. Which escape sequence is used for:

- New line
- Tab
- Backslash
- Audible bell
- Carriage return

Answer:

Q29. What is the purpose of comments in C?

Q30. Write a block comment that spans two lines.

Q31. What is the syntax of `printf` and `scanf`?

Q32. What does the format specifier `%d` represent?

Q33. What is the role of the `&` symbol in `scanf`?

Q34. What is the structure of a C program?

Q35. What is a token in C programming? List the six types of tokens.

Q36. Define a keyword and explain why its meaning cannot be changed.

Q37. What is an identifier in C? List two rules for naming identifiers.

Q38. Explain the difference between valid and invalid identifiers with examples.

Q39. What is a data type in C and why is it important?

Q40. Define the `float` data type and describe its precision and size.

Q41. What is the difference between signed and unsigned `char` types?

Q42. What is a constant in C? How does it differ from a variable? Include examples using `const`.

Q43. Define an escape sequence and give two examples with their meanings.

Q44. What does the `scanf()` function do in C? How is it different from `printf()`?

Q45. Compare machine language, symbolic language, and high-level language in terms of readability, hardware compatibility, and ease of use.

Q46. Why is C considered a structured programming language? What advantages does this offer to developers?

Q47. What are the steps involved in creating and running a C program? Briefly describe each phase.

Q48. How does the compiler differ from the linker in the program development cycle?

Q49. What does the error loop represent in the context of compiling and debugging?

Q50. Identify which of the following are valid identifiers and explain why: `student_name`, `2names`, `sum-salary`, `INT MIN`.

Q51. What are the consequences of using a keyword as an identifier in a C program?

Q52. Why is C considered a case-sensitive language? Provide examples to illustrate.

Q53. Provide examples of integer, real, character, and string constants. What syntax rules apply to each?

Q54. Match each data type with its size, range, and precision:

Data Types	Attributes
a. `char`	1. 1 byte, range: -128 to 127
b. `int`	2. 4 bytes, precision: 6 digits
c. `float`	3. 8 bytes, precision: 15 digits
d. `double`	4. 2 or 4 bytes, range: platform-dependent

Your Matches: (Write the letter next to the correct number)

Q55. Match each format specifier with its corresponding data type:

Format Specifiers	Data Types
a. %d	1. float
b. %f	2. int
c. %c	3. char
d. %lf	4. double

Your Matches: (Write the letter next to the correct number)

_____

_____

_____

Q56. Match each software tool with its role in program development:

Tools

a. Compiler

b. Text Editor

c. Linker

d. Runner

Roles

1. Converts source code to object code
2. Allows writing and editing source code
3. Combines object files into a single executable
4. Executes compiled programs
5. Helps find and fix errors

Your Matches: (Write the letter next to the correct number)

_____

_____

_____

Q57. Match each component with its definition:

Components

a. Compiler

b. Source Program

c. Executable Program

d. Preprocessor Directive

e. main() Function

Definitions

1. The entry point of every C program.
2. A set of instructions written in a high-level language.
3. A translated version of the source code that can be run by the computer.
4. A tool that converts source code into machine code.
5. A command that instructs the compiler to include or modify code before compilation.

Your Matches: (Write the letter next to the correct number)

Q58. Match each token type with its description:

Token Types

a. Keyword

b. Identifier

c. Constant

d. Operator

e. Separator

f. String Literal

Descriptions

1. A reserved word with a predefined meaning in C.
2. A name used to identify variables, functions, arrays, etc.
3. A fixed value that does not change during program execution.
4. A symbol that performs operations on variables and values.
5. A character used to separate statements or elements.
6. A sequence of characters enclosed in double quotes.

Your Matches: (Write the letter next to the correct number)

Q59. Match each escape sequence with its meaning:

Escape Sequences	**Meanings**
a. `\n`	1. Inserts a tab space.
b. `\t`	2. Inserts a newline.
c. `\r`	3. Inserts a carriage return.
d. `\"`	4. Inserts a double quote character.

Your Matches: (Write the letter next to the correct number)

_____

_____

_____

Q60. Fill in the blanks with the correct term:

- A _____ is the only language directly understood by computer hardware.
- The directive `#include <stdio.h>` is an example of a _____.
- In C, comments that span multiple lines are called _____ comments.
- A _____ is a meaningful sequence of characters used to name variables, functions, etc.
- The function used to display output in C is _____.
- The sequence of instructions written to perform a specific task is called a _____.
- These instructions are formed using special symbols and words according to rigid rules known as _____.
- Every instruction in C must follow the language's _____ rules.
- Like any other language, C has its own _____ and _____.
- A program written in a high-level language is called a _____.
- The tool used to translate a source program into machine language is called a _____.
- The translated version of a source program is called an _____ program.

- The process of combining object code with library code is called _____.
- The only language understood by computer hardware is _____ language.
- Every C program must have one function named _____.
- The directive `#include <stdio.h>` is an example of a _____ directive.
- The keyword `void` means the function _____ return any value.
- Statements inside the `main()` function are enclosed within _____.
- In C, the smallest individual units are called _____.
- There are six types of tokens: _____, _____, _____, _____, _____, and _____.
- An identifier must begin with a _____ or an underscore.
- Identifiers cannot contain _____ or _____.
- C is a _____-sensitive language.
- The four basic data types in C are _____, _____, _____, and _____.
- A variable is a named _____ location that stores a value.
- The assignment operator in C is _____.
- A character constant is enclosed in _____ quotes, while a string constant is enclosed in _____ quotes.
- The keyword `const` is used to declare a _____ whose value cannot change.
- The function `printf()` is used for _____ output.
- The function `scanf()` is used for _____ input.
- The symbol `&` is called the _____ operator and is used to pass the address of a variable.
- The escape sequence `\n` inserts a _____.
- The escape sequence `\t` inserts a _____.

Valid Names		Invalid Name	
a	// Valid but poor style	$sum	// $ is illegal
student_name		2names	// First char digit
_aSystemName		sum-salary	// Contains hyphen
_Bool	// Boolean System id	stdnt Nmbr	// Contains spaces
INT_MIN	// System Defined Value	int	// Keyword

Figure 1: Examples of Valid and Invalid Names

Q61. Using the table of valid/invalid identifiers: Choose two invalid identifiers and explain why they are incorrect.

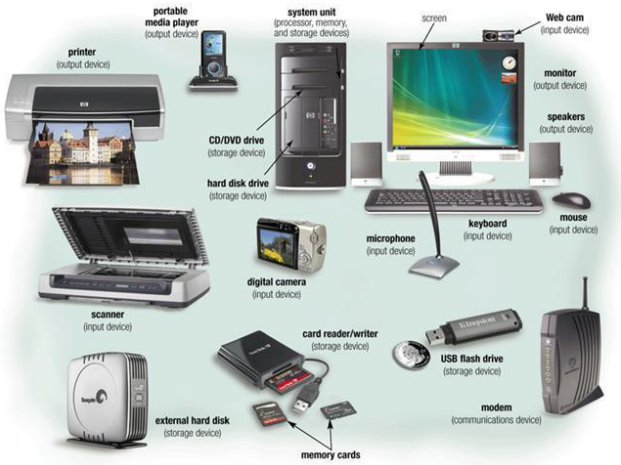

Figure 2: Basic Hardware Components

Q62. Referencing the image of the computer hardware components: Identify and define three input devices and three output devices shown in the diagram.

Section 2: Trace the Code – Predict the Output

T1.
```
int a = 5, b = 2;
int result = a * b + a / b;
printf("%d", result);
```

Output:

T2.
```
int x = 10;
x = x + 5;
x = x - 3;
printf("Final value: %d", x);
```

Output:

T3.
```
char ch = 'A';
printf("%c %d", ch, ch);
```

Output:

T4.
```
int num = 42;
printf("%5d", num);
```

Output:

T5.
```
float pi = 3.14159;
printf("%.2f", pi);
```

Output:

T6.
```
printf("Hello\tWorld\nGoodbye\rMoon");
```

Output:

T7.
```
int a, b;
scanf("%d%d", &a, &b);
// Input: 3 4
printf("%d", a + b);
```

Output:

T8.
```
int x = 7, y = 2;
printf("%d", x / y);
```

Output:

T9.
```
int a = 1, b = 2, c = 3;
int total = a + b + c;
printf("Sum = %d", total);
```

Output:

T10.
```
int x;
printf("%d", x);
```

Expected behavior:

Section 3: Debug the Code – Fix the Errors

D1.
```
int a = 5
printf("%d", a);
```

Fix:

D2.
```
int x = 10;
y = x + 5;
printf("%d", y);
```

Fix:

D3.
```
int age = 25;
printf("Age: %f", age);
```

Fix:

D4.
```
printf("Hello World);
```

Fix:

D5.
```
int num;
scanf("%d", num);
```

Fix:

D6.
```
int result = 5 / 2;
printf("%f", result);
```

Fix:

D7.
```
int x;
printf("%d", x);
```

Fix:

Section 4: Extra Questions

C1.
```c
#include <stdio.h>

int main() {
    printf("Hello, World!\n")
    return 0;
```

Fix:

C2.
```c
#include <stdio.h>

int main() {
    / This prints a message
    printf("Learning C is fun!\n");
    return 0;
}
```

Fix:

C3.
```c
int age = 25;
float height = 5.9;
char grade = "A";

printf("Age: %f\n", age);
printf("Height: %d\n", height);
printf("Grade: %s\n", grade);
```

Fix:

C4.
```c
int number;
printf("Enter a number: ");
scanf("%d", number);
printf("You entered: %d\n", number);
```

Fix:

C5.
```
printf("Line1/nLine2/tTabbed\nBackslash:-\");
```

Fix:

C6.
```
int a = 5, b = 2;
int result = a * b + a /;
printf("%d", result);
```

Fix:

C7.
```
int x = 150;
char a = 'A';
float pi = 3.14;
printf("%f-%d-%c", x, a, pi);
```

Fix:

End of Quiz — Great work!